FYS-STK4155 - Project 1

Geir Tore Ulvik, Idun Klvstad

October 13, 2018

Abstract

This project looks at three different regression methods, Ordinary Least Squares, Lasso and Ridge, used on a known function together with the bootstrap resampling method. It compares the three methods by looking at the Mean Squared error and the R^2 score. The results show that the OLS method performs the best when trying to estimate Franke's function.

1 Introduction

This report focuses on regression analysis and resampling methods. Three different regression methods will be combined with the bootsrap method and compared. The regression models that will be looked at are the Ordinary Least Squares (OLS) method, Ridge regression and Lasso regression.

The first thing we will study is how to fit polynomials of different order to a spesific two-dimensional function called Franke's function. Thereafter the same methods will be used, trying to reproduce digital terrain data.

The report is structured so that there is first given a bit of theory about the different methods, then we will present how we chose to implement those methods and why, before we show and discuss the results of using the different methods and finally makes a conclusion of what we think are each methods strong and weak sides.

2 Theory

2.1 Franke's function

Franke's function is a weighted sum of four exponentials and has been widely used when testing various interpolation and fitting algorithms. The function is given by

$$f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right).$$

. [1]

2.2 Linear methods for regression

A linear model assumes that the regression function is linear in the inputs $X_1, \ldots, X_p$, and can give a polynomial representation using basic expansions such as $X_2 = X_1^2, X_3 = X_1^3$.

If we have an input vector as described above and want to predict a real-valued output Y, the linear model reads as

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{1}$$

[2]

For all three regression methods that we will look at in this report, there is a set of training data from which we want to estimate the parameter β.

2.2.1 Ordinary Least Squares method

The Ordinary Least Squares (OLS) method pick the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ to minimize the residual sum of squares given by

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2 \tag{2}$$

$$= \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2 \tag{3}$$

$$\tag{4}$$

This can be rewritten with matrix notation

$$RSS(\beta) = (\hat{y} - \vec{X}\beta)^T(\hat{y} - \hat{X}\beta) \tag{5}$$

. [2] As this is explained in detail in the book as well as in the lecture notes [3], we will not derive it here.

As we want to minimize the residual sum of squares, we require

$$\frac{\partial RSS(\hat{\beta})}{\partial \hat{\beta}} = 0 = \hat{X}^T(\hat{y} - \hat{X}\hat{\beta}) \tag{6}$$

This can be rewritten as

$$\hat{X}^T\hat{y} = \hat{X}^T\hat{X}\hat{\beta} \tag{7}$$

.

We now has to assume that the matrix $\hat{X}^T\hat{X}$ is invertible to get the solution

$$\hat{\beta} = (\hat{X}^T\hat{X})^{-1}\hat{X}^T\hat{y} \tag{8}$$

[3] [2]

2.2.2 Ridge regression

The OLS will not have a solution if the design matrix $\hat{X}$ is singular or near singular, as it depends in $\hat{X}^T\hat{X}$ being invetertible. [3]

Ridge regression is known as a shrinkage method. It shrinks the regression coefficients $\hat{\beta}$ by imposing a penalty on their size [2]. We are in other words adding a diagonal component to the matrix to invert. From OLS we therefore make the following change

$$\hat{X}^T\hat{X} \rightarrow \hat{X}^T\hat{X} + \lambda\hat{I} \tag{9}$$

where I is the identity matrix. [3]. Thus we get

$$RSS(\lambda) = (\hat{y} - \hat{X}\hat{\beta})^T(\hat{y} - \hat{X}\hat{\beta}) - \lambda\hat{\beta}^T\hat{\beta} \tag{10}$$

and

$$\hat{\beta}^{ridge} = (\hat{X}^T\hat{X} + \lambda\hat{I})^{-1}\hat{X}^T\hat{y} \tag{11}$$

$\lambda > 0$ is a complexity parameter that controls the amount of shrinkage. The larger calue of λ the greater amount of shrinkage. [2]

This can also be written in a non-vector format which makes explicit the size constraint on the parameters. It also makes it easier to see the difference from Lasso regression:

$$\hat{\beta}^{ridge} = \text{argmin}_{beta}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 \tag{12}$$

subject to

$$\sum_{j=1}^{p}\beta_j^2 \leq t \tag{13}$$

There is a one to one correspondence between λ and t. [2]

2.2.3 Lasso regression

Like ridge, lasso is a shrinkage method. The main difference from ridge is that lasso has no closed form expression. [2]

Similarly to ridge, the lasso estimate is given by

$$\hat{\beta}^{lasso} = \text{argmin}_{beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \tag{14}$$

subject to

$$\sum_{j=1}^{p} |\beta_j| \leq t \tag{15}$$

.

As you can see, the ridge penalty $\sum_1^p \beta_j^2$ is replaced by a lasso penalty given by $\sum_1^p |\beta_j|$. This makes the solution nonlinear in the y_i and is the reason that the lasso solution don't have a closed form expression.

Lasso does a kind of continous subset selection because of the nature of the constraint. The reason for that is that making t sufficiently small, will cause some of the coefficients to go to zero. [2]

2.3 Error estimates

To evaluate the different regression methods we have two main measures that can be used. In both equation below $\widetilde{y}_i$ is the predicted value of the i-th sample and that y_i is the corresponding true value. [1] The first is the Mean Squared Error (MSE) which is given by

$$MSE(\hat{y}, \widetilde{\hat{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i, \widetilde{y}_i)^2 \tag{16}$$

. The other is the R^2 score which is defined by

$$R^2(\hat{y}, \tilde{\hat{y}}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \tilde{y} y_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2}, \tag{17}$$

where we have defined the mean value of $\hat{y}$ as

$$\bar{y} = \frac{1}{n} \sum_{i=0}^{n-1} y_i. \tag{18}$$

[1]

2.4 Resampling methods

A resampling method is a tool that involves repeatingly drawing samples from a training data set and refitting a model of interest on each sample, in order to obtain additional information about the fitted model. The reason for using a resampling method, is to obtain information that would not be available from fitting the model only once using the original training sample. [3]

2.4.1 Bootstrap

Bootstrap is a resampling method, and is widely used. The bootstrap method is based on the fact that $\hat{\Theta} = \hat{\Theta}$ is a random variable, because it is a function of random variables. Therefore it has a pdf, $p(\vec{t})$. The aim of the bootstrap is to estimate $p(\vec{t})$ by the relative frequency of $\hat{\Theta}$ [3]

The bootstrap method works with four different steps:

1. Draw with replacement n numbers for the observed variables $\hat{x} = (x_1, x_2, \cdots, x_n)$.
2. Define a vector $\hat{x}^*$ containing the values which were drawn from $\hat{x}$.
3. Using the vector $\hat{x}^*$ compute $\hat{\Theta}^*$ by evaluating $\hat{\Theta}$ under the observations $\hat{x}^*$.
4. Repeat the process k times.

[3]

3 Implementation of methods

All code can be found on github, using the following link:
https://github.com/geirtul/fys-stk4155/tree/master/project1/src

There is one class for each regression method, with corresponding filenames. All the error estimations are done in analysis.py

In all the methods, Franke's function is defined for $x, y \epsilon [0, 1]$.

We have not written unittests for the code. We are very aware that that would have been a good idea.

3.1 Ordinary Least Squares

The class ols.py i made to make a prediction of Franke's function based on the OLS method. It makes x, y as arrays with uniformly distributed random numbers between 0 and 1.

It uses scikit learn to make a polynomial of a given degree.

The following lines calculates $\hat{\beta}$ as described in section 2.2.1

```
        # Regression
        X = self.poly.fit_transform(self.predictors)
        self.beta = np.linalg.inv(X.T @ X) @ X.T @ outcome
```

Then the following lines makes a predicted outcome:

```
        X = self.poly.fit_transform(x_in)
        self.predicted_outcome = X @ self.beta
        self.outcome = z_in
```

3.2 Ridge

As stated in section 2.2.2, there is not a huge difference from OLS to ridge regression.

The code for calculating $\hat{\beta}$ from the math shown in the section looks like this:

```
        # Regression
        X = self.poly.fit_transform(self.predictors)

        I = np.eye(len(X[1]))

        self.beta = (np.linalg.inv(X.T @ X + lmb * I) @ X.T @
           outcome)
```

.

As we only change β, the code for finding the estimated function is the same, only with the new beta.

3.3 Lasso

We chose to use scikit learn for the lasso regression. The code should do what is described in 2.2.3.

The coefficients is therefore calculated like this

```
        # Regression
        # Input values to design matrix
        self.predictors = self.poly.fit_transform(predictors)

        self.lasso_object = linear_model.Lasso(alpha=self.alpha,
           max_iter=1e3)
        self.lasso_object.fit(self.predictors, self.outcome)
```

.

and the predicted function like this

```
        X = self.poly.fit_transform(x_in)
        self.predicted_outcome = self.lasso_object.predict(X)
        self.outcome = z_in
```

.

3.4 Analysis

The analysis class contains all of the error calculations as well as the bootstrap method and a method for making plots.

As the error calculation is straight forward math, we do not show the implementation in the report.

3.4.1 Bootstrap

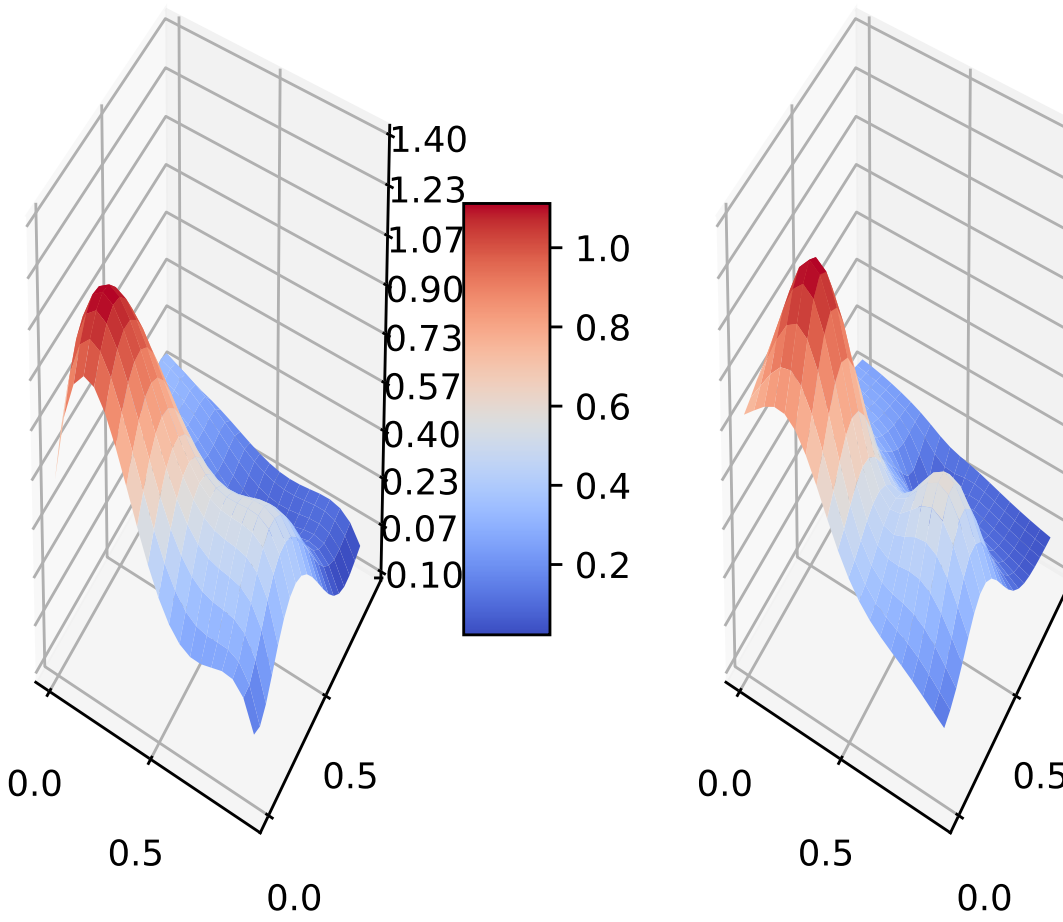
We chose to use bootstrap because of a number of advantages listed in the lecture notes. [3]

The data is first split into training and test sets. Then the bootstrap method described in **??** is implemented as follows

```
        for i in range(n_bootstraps):
            x_re, data_re = resample(x_train, data_train, replace=
               True)
            self.fit_coefficients(x_re, data_re, self.poly_degree)
            newfit = self.make_prediction(x_test, data_test)
            y_fits[i] = newfit.ravel()
```

. [4]

4 Results


1.40
1.23
1.07
0.90
0.73
0.57
0.40
0.23
0.07
0.10
1.0
0.8
0.6
0.4
0.2
0.0
0.5
0.0
0.5
0.0
0.5
0.0
0.5

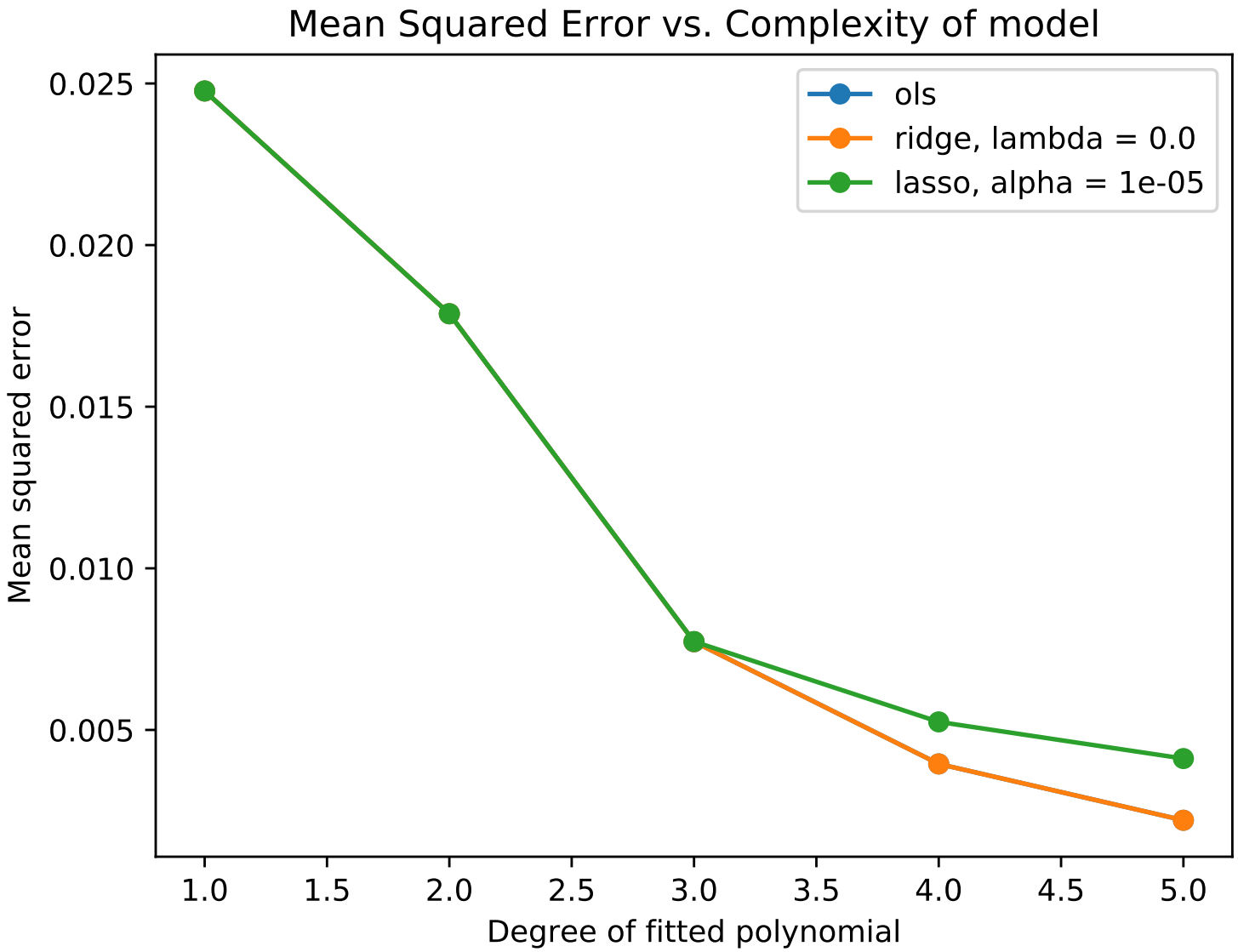


Figure 1: Plot of Mean Squared Error as a function of model complexity (degree of the fitted polynomial). ols is ordinary least squares regression. The data used in this plots is the testing data generated by the Franke Function.

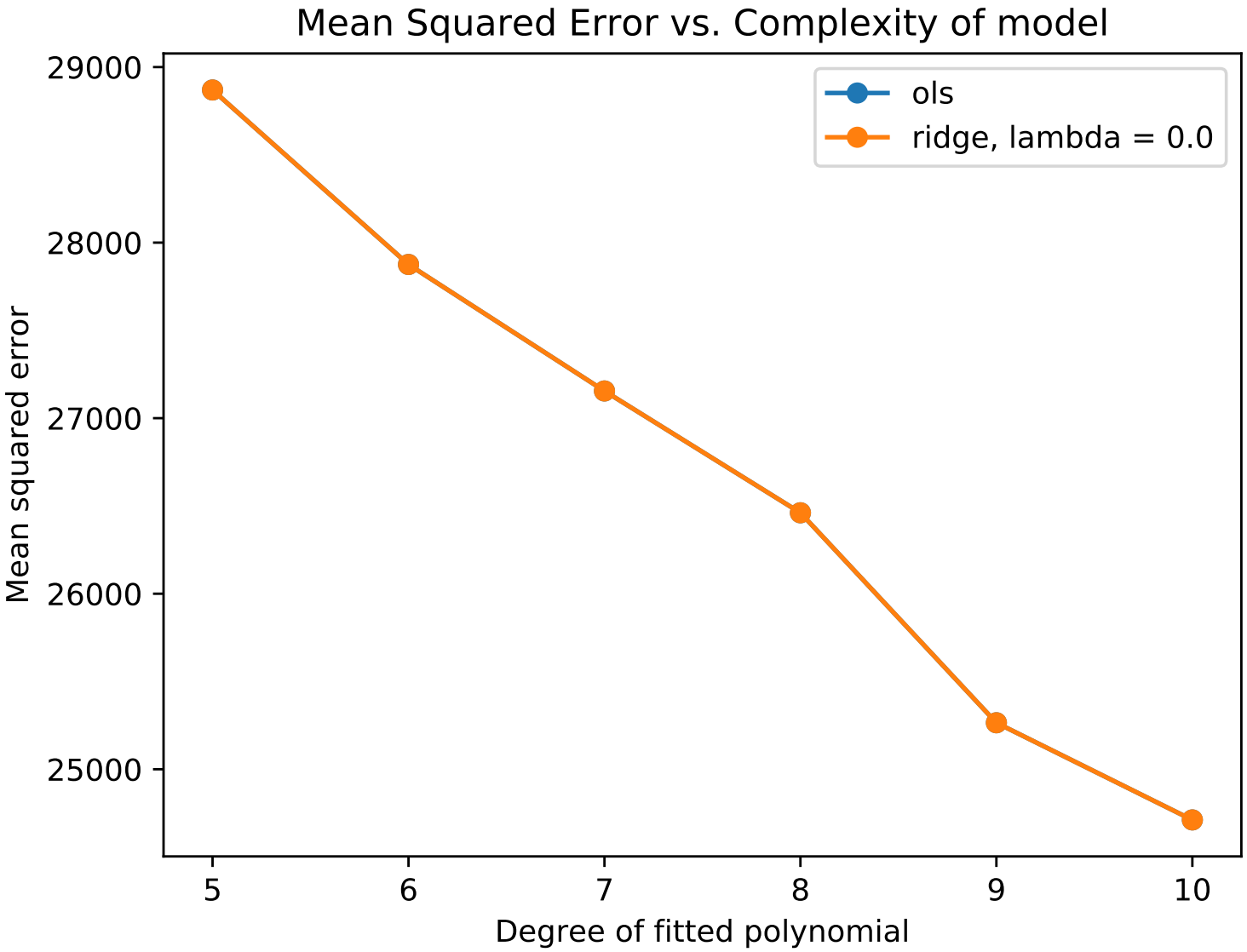


Figure 2: Plot of Mean Squared Error as a function of model complexity (degree of the fitted polynomial). ols is ordinary least squares regression. Regression was performed on real terrain data from a region in Norway.

5 Discussion

The plot in figure red1Plot of Mean Squared Error as a function of model complexity (degree of the fitted polynomial). ols is ordinary least squares regression. The data used in this plots is the testing data generated by the Franke Function.figure.caption.1 show the MSE declining with model complexity, as would be expected for the Franke Function dataset, since it actually includes a polynomial of degree 5. As can be seen from the visualization in **??** this is indeed the case, as the surface imitates that of the dataset quite well, but not perfectly. For the real-world terrain data, the MSE follows a similar trend, shown in figure

6 Conclusion

References

[1] University of Oslo Department of Physics. Project 1 on machine learning. magentahttps://compphysics.github.io/MachineLearning/doc/Projects/2018/Project1/html/Project1-bs.html, 2018.

[2] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[3] M Hjorth-Jensen. Data analysis and machine learning: Linear regression and more advanced regression analysis - lecture notes. magentahttps://compphysics.github.io/MachineLearning/doc/pub/Regression/html/._Regression-bs000.html, 2018.

[4] Bendik Samseth. Piazza post on bias and variance. https://piazza.com/class/ji78s1cduul39a?cid=59, 2018.